

Modeling Resource-Coupled Computations

Mark Hereld,^{†*} Joseph Insley,[†] Eric Olson,^{*} Michael E. Papka,^{†*}
Thomas Uram,[†] Venkatram Vishwanath[†]

[†]Argonne National Laboratory, ^{*}The University of Chicago

{hereld, insley, eolson, papka, turam, venkatv}@mcs.anl.gov

ABSTRACT

Increasingly massive datasets produced by simulations beg the question How will we connect this data to the computational and display resources that support visualization and analysis? This question is driving research into new approaches to allocating computational, storage, and network resources. In this paper we explore potential solutions that couple system resources in new ways.

Examples of what we mean by resource-coupled computations abound. For example, remote visualization is an activity that may couple data and large computation resources at the shared facility to client software and display hardware at the remote site. In situ analysis and visualization contemporaneously merges simulation and analysis onto the shared resource of the supercomputing platform. Co-analysis approaches seek to directly couple simulations running on a primary supercomputer to live analysis running on an optimized visualization and analysis platform over a high-performance network.

Consequently, we are working on a systems approach to modeling the end-to-end activity of extracting understanding from computational models. In this paper we present our methods and results from experiments.

Categories and Subject Descriptors

C.0 [Computer System Organization]: General – *modeling of computer architecture, system architectures*. C.2 [Computer Systems Organization]: Computer-communication networks – *data communications*. C.4 [Computer Systems Organization]: Performance of systems – *design studies, measurement techniques, modeling techniques, performance attributes*. D.4.8 [Operating Systems]: Performance – *measurements, modeling and prediction*.

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

High-performance computing, data intensive computing, coupled computations, simulation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION

Large-scale computational science requires substantial hardware resources for simulation, data storage, and data analysis. Current systems supporting National Science Foundation (NSF) and Department of Energy (DOE) scientific programs include configurations at the National Center for Supercomputing Applications (NCSA), the Texas Advanced Computing Center (TACC), the National Institute for Computational Sciences (NICS), Argonne National Laboratory (ANL), Oak Ridge National Laboratory (ORNL), and Lawrence Berkeley National Laboratory (LBNL). The massively parallel computational components are supported by parallel file systems capable of moving several tens of gigabytes per second and are connected to global backbone networks that can carry about a tenth of that traffic.

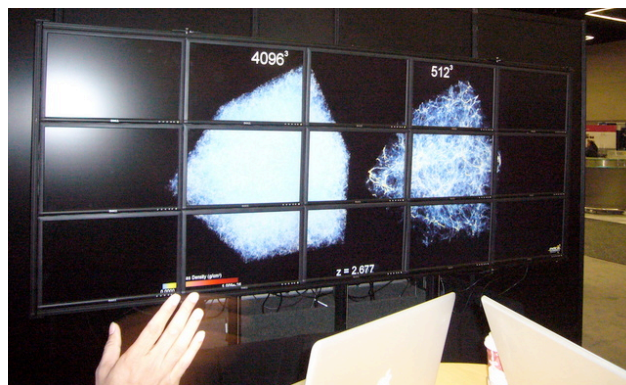


Figure 1. Live streaming of 10 Gbps from Argonne National Laboratory to Portland, OR, at Supercomputing 2009.

The problem of analyzing datasets produced by large simulation has spawned a number of ideas. The traditional approach to data analysis and visualization, namely, postprocessing, finds its data in local or networked storage, perhaps distributed widely across many storage resources. Postprocessing is what is commonly thought of when discussing visualization and analysis. Postprocessing happens after the simulation has completed or at least after data has been written to disk. Much of the visualization research done within the community is focused on use within a postprocessing application. The postprocessing stage in the pipeline has been packaged into many well-known environments such as Matlab, Mathematica, and R, as well as more HPC-oriented environments such as VisIt [<http://www.llnl.gov/visit/>] and ParaView [<http://www.paraview.org/>]. But issues of data scale and the cost of staging the data to and from disk between creation and analysis steps have prompted research into other

strategies. In situ approaches take the view that the data will never be more available than it is at the moment it is created by the simulation. Other research aims to insert analysis in tandem with the I/O operations required to save data to disk. With tightly coupled computational systems one can consider real-time co-analysis with data shipping across the high-performance interconnect.

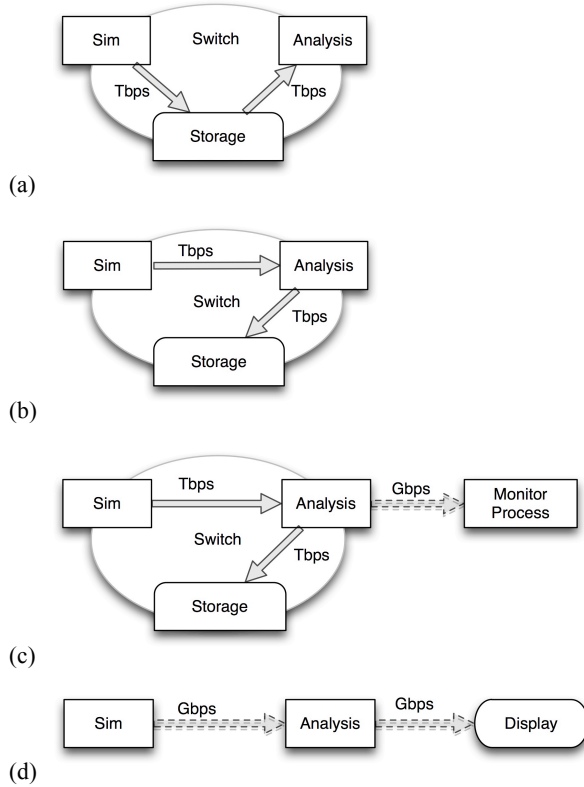


Figure 2. Some of the many useful computational pipeline configurations: (a) nominal pipeline configuration, (b) simulation process communicates directly over the facility network with optimized analysis hardware, (c) monitoring clients attached to the pipeline, and (d) high-performance computational science on resources spread over wide area network.

We define in situ analysis as analysis and/or visualization code that runs as part of the simulation code at the same time as the simulation. This approach is receiving much attention within the high-performance computing community as perhaps the only way to analyze future-generation large-scale datasets. An example of in situ analysis can be found in the efforts Yu et al. [10], and an overview is presented by Ma [8]. Also relevant to the algorithm performance piece of the in situ problem is the long train of work in scaling parallel visualization algorithms. These analyses and algorithms have not yet been applied in situ, however; and much work remains to be done to characterize co-resident performance with a running simulation.

We define co-analysis as processing that runs in parallel with the simulation but not necessarily on the same resources or even at the same site. CUMULVS [6] is an example of a co-analysis

approach that provides means for interacting with and steering a running parallel simulation. LambdaRAM [5][9] enables data coupling between clusters interconnected via high-speed optical networks in order to facilitate co-analysis. LIVE [1] provides a means for connecting to a running simulation in order to monitor or analyze data generated by the application in real time. Also related is work aimed at connecting separated resources [2][3][4] or work such as the Model Coupling Toolkit, which enables construction of multiphysics, parallel coupled models [7].

Figure 2 shows many of the current and envisioned pipeline configurations that might support high-performance coupled-resource computation. The nominal pipeline configuration in Figure 2(a) illustrates the case wherein the simulation stores results to disk and an analysis application works from disk. Ideally, all communication is carried out over very high speed switch fabric. When the simulation process can communicate directly over the facility network with nominal bandwidth of terabits per second, advanced analysis can be performed on optimized hardware, as depicted in Figure 2(b). The desirable but-challenging configuration shown in Figure 2(d) enables high performance computational science on resources spread over a wide-area network. A demonstration of this, shown in Figure 1, is described later in the text.

Two main questions drive the research:

- Where in the pipeline is the right place to do analysis and visualization?
- Will these capabilities improve time-to-discovery?

These lead to a number of corollary questions that focus on particular scenarios, hypotheses, and issues, including the following:

- Is it better to carry out some of the analysis in situ with the simulation or to move the data to a separate analysis process?
- What are the circumstances where we can speed application by offloading I/O quickly, and how often do they arise?
- Under what circumstances can wide-area co-analysis be effective given the lower bandwidth of external network?

The effort described in this paper touches many portions of the computational science pipeline. Before we can answer these bigger, more integrative questions, we need to understand the cost of performing data analysis or partial analyses at the various points in the pipeline—in terms of time, bytes, and dollars. How we are attacking this question is the subject of this paper.

To get answers, we are designing models at a coarse granularity that enable high-level assessment of opportunities and potential benefits. In the following sections, we present our preliminary efforts to gather and analyze data from a number of sources before digging into building models, designing experiments, and testing measurements against models.

2. Multipronged Approach

In this section we describe our work in progress to measure, characterize, and model current supercomputing resources in order to understand how future ultrascale systems will be used.

We target the idealized configurations of the computational pipeline shown in Figure 2 in terms of the basic properties of the hardware, application software, and analysis software components of the system.

2.1 About Applications

Understanding an application's run-time behavior is critical in order to gain insight into when and where analyses might be performed. Key parameters include the application's computational requirements, its memory usage, and the data movement patterns.

The amount of data emitted by a given application per time step, per data type, and/or per checkpoint cycle will provide the basis for estimating time-dependent memory requirements for our models.

We want to understand how in situ approaches compare to co-analysis or how much automatic analysis and of what sort can be overlapped with the running simulation. For this investigation we will need estimates of how expensive the relevant computations (both in the simulation and in the analysis) are on each of the computational resources.

A key determinant of the level of success achieved by these methods will be the extent to which data can be moved quickly from the simulation to the analysis. This will rely on the amount of raw network bandwidth available and the effectiveness with which it is used.

For some application scenarios, latency will be an important parameter in our models; we are thinking here of interactive applications or those with some monitoring. Its importance in other scenarios will depend on the granularity of the elements in the data stream and to what extent data must be reorganized across the cluster nodes.

Coarse-grained data patterns will provide critical insight into which parts of the data stream are most constraining and where opportune windows provide space for overlapped analysis. We will capture aspects of the spacing, data bulk, and data rates in our models.

2.2 About Resources

Large-scale scientific applications have demonstrated scalable performance on today's architectures. However, based on trends extracted from available market literature (as listed in Table 1), we can expect that future architectures will require that today's applications and middleware scale their performance in ways previously unexplored. Key trends in future architectures include the large number of processing cores on a chip as well as the increase in the network bandwidth throughout the various subsystems. Energy efficiency is another key criterion listed in Table 2 for future systems. Thus, a critical component for scalable performance will be the development of new techniques for efficient utilization of end-system architectures and resources.

For our models and experiments to be most beneficial, we will organize them around a set of generalized architectures including mixtures of high-performance computing platforms, visualization and

analysis clusters, parallel file systems, high-performance switching interconnection hardware, and wide-area networks. This approach will enable us to consider tightly coupled systems as well as loosely coupled federations of systems.

Table 1. Trends in bandwidth. Data compiled from the released products of various hardware vendors and their future projected roadmaps.

Bandwidth Trends (Gbps)	2007	2009	2011
Disk Drive	1	3	6
Network Interface	10	30	100
Memory	100	40	384
GPU Memory	500	1000	2000
Processor Interconnect	51	120	200
Wide Area	10	40	100

Table 2. Trends in node architecture. Data compiled from the released products of various hardware vendors and their future projected roadmaps.

	CPU			GPU		
	2007	2009	2011	2007	2009	2011
State of the Art	Opteron	Nehalem	Larabee	Nvidia 8800	ATI 4870	ATI 6800
Core per Processor	4	8	12	256	800	4000
Computational Efficiency GFlops / Proc	16	54	200	256	1000	4000
Energy Efficiency GWatts / Proc	0.16	0.5	2	2.5	10	40

The primary resources considered in the model will include the state-of-the-art HPC systems currently deployed, in order to (1) allow us to test the widest range of performance parameters, (2) prototype useful solutions to current problems, and (3) make more accurate extrapolations to future hardware systems. The resources include those at the Argonne Leadership Computing Facility (ALCF). The principal components of the system to be used as a

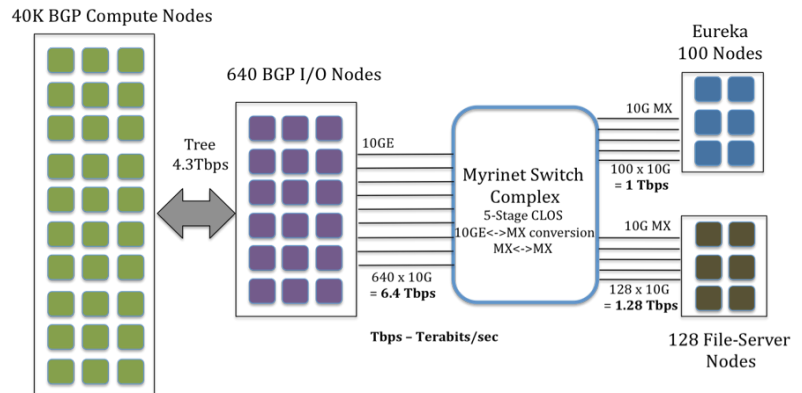


Figure 3. Architecture of the coupled resources in the Leadership Computing Facility at Argonne National Laboratory.

generic testbed are shown in Figure 3. Intrepid is a 160K core IBM Blue Gene/P system with peak performance of 557 TF, 80 TB local memory, and 640 I/O nodes each with 4 cores, connected to the switching interconnect with aggregate 6.4 Tbps. Eureka comprises 100 servers with 800 Xeon cores, 3.2 TB memory, and 200 nVidia Quadro FX 5600 GPUs. Eureka is connected to the switch with 100 links at 10 Gbps each. The file server is connected to the switch with 128 links at 10 Gbps each.

Models and test results on this system will provide insight into performance of the co-analysis techniques to be studied for a range of current systems, as well as a handle on how future systems might perform. First, the high-end system is in some sense a proxy of more modest systems of the future. The terabit bandwidth of the switching fabric will represent global backbone bandwidths in coming years. Second, experiments run on this system can be throttled at various points to reflect expected performance for a range of currently available systems.

2.3 About Analysis

Our goals in understanding the run-time behavior of analysis and visualization codes is analogous to our interest described in application codes.

Several widely used data analysis and visualization applications serve the scientific HPC community. These include VisIt and ParaView, which can execute large-scale parallel visualizations; Matlab; R; and a number of more specialized, domain-tailored packages. Some researchers find that their analysis and visualization can be done only by customized applications. These are sometimes implemented by using high-level toolkits such as vtk (<http://www.vtk.org/>), the visualization toolkit that is the basis for both ParaView and VisIt. An example of an even more homegrown code is our own volume-rendering application, v13, which exploits GPU hardware to speed evaluation and accumulation of transfer function modulated data along cast rays. This is accomplished with custom GPU code.

We seek to understand how these examples behave on a range of platforms. Hence, we will use modeling techniques similar to those used in science application codes.

3. Current Efforts

Here we describe our current works in characterizing resource usage by science and analysis applications, detailed performance and scaling properties of algorithms, hardware performance characteristics, and performance of computational science on widely distributed resources.

3.1 Footprint Measurements

We want data about the properties and mix of jobs running on Intrepid and Eureka, including data about how these jobs use the storage system. From this data we will be able to understand better what kinds of footprints typical and extreme science problems leave on the mix of resources in a high-performance computing facility: simulation cycles consumed by runs and campaigns, amount and creation patterns of data produced by simulations, postprocessing access patterns for analysis and visualization, and cycles consumed in data exploration and reduction. This data is not widely available and has not been systematically collected and studied.

Toward this goal, we have collected into databases information available from system logs on Intrepid and Eureka. The automating scripts for this ongoing metadata collection are in place and operating. We are mining this information for statistics, trends, and norms to help guide our modeling and design activities.

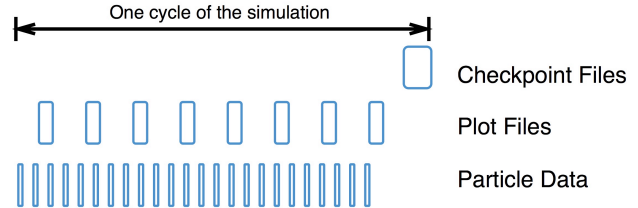


Figure 4. Example showing the high level I/O pattern of one particular astrophysics simulation. Three separate types of files are written at different intervals over the course of the simulation cycle: particle files, plot files, and checkpoint files. Shown is one cycle ending with a checkpoint.

We have begun work to understand how one simulation using a multiphysics code might benefit from a co-processing approach. Consider routine simulations of Type Ia supernovae currently being run on Intrepid as part of the DOE INCITE program. An example of coarse-grained data patterns is shown in Figure 4. The code includes modules for gravitation, nuclear chemistry, and magnetohydrodynamics. It is highly configurable and can be used to solve problems from laboratory-scale to cosmology; hence, performance characteristics vary greatly from problem to problem.

These simulations are generally run over a period of several days to a week in twelve-hour jobs. Each run is marked at intervals of about 90 minutes by a checkpoint that enables the job to be restarted from that point in the event of system failure; it also provides much of the data used in the subsequent analysis. In addition to these checkpoint files, the application writes particle and plot files containing different kinds of data. Figure 4 shows the pattern of these data-writing operations.

Table 3. Example data footprint.

File Type	File Size	#files / Run	#files / Sim	Data Size
Particle	~ 131 MB	~ 500	5000	500 GB
Plot	~ 13 GB	40-90	800	10 TB
Checkpoint	~ 42 GB	5-10	100	4.2 TB

Table 3 fills in the details of actual file sizes and numbers for each of the file types generated. In this case the particle file writes take a disproportionate fraction of the I/O time despite the relatively small fraction of the dataset invested in them. These blocking I/O operations take as much as 30 percent of the total execution time. The numerous particle files generate 500 GB over the course of the simulation and take a disproportionate fraction of the overall I/O time but are amenable to optimizations not yet implemented in the simulation. The plot files contain the bulk of the data written

during the simulation, some 10 TB, while the checkpoint files produce almost a third of the total.

From this information we see several opportunities that might be exploitable. First, to the extent that the I/O takes more time than available bandwidth would imply, we might improve overall application execution time if we can offload the I/O to another compute resource. Second, this same system might provide value added by overlapping analysis with the running simulation.

3.2 Application Performance Measurements

We are studying the performance characteristics of v13, mentioned briefly above, because it can take unique advantage of the GPU. Hence, we can study in detail the possible advantages of specialized hardware acceleration with purely software rendering. By modeling it and testing in on Intrepid (without the aid of the acceleration) and on Eureka, we gain insight into the tradeoffs between hardware and software execution. We have instrumented the code using the same algorithm without the benefit of hardware acceleration.

The parallel program casts rays through a very large data volume, evaluating a transfer function as it accumulates translucent coloration depending on the sampled data values. Compositing the results from all processors into a single image requires data exchange over the network. The resulting frame can be written to disk or sent to a remote display system by using as many parallel streams as necessary.

To date, we have conducted a number of experiments with v13. For example, we have performed scaling measurements on Eureka using the GPU acceleration of the ray casting with 128 processes running on 64 nodes. On this machine we have been able to render data cubes as large as 6K on a side. We have also performed scaling measurements with v13 on Intrepid, casting rays in software. Furthermore, we have run v13 in an interactive test of high-performance rendering from a remote site over a wide-area network (described below).

3.3 Internal Network Experiments

We are also characterizing performance of the new and unconventional data path between Intrepid and Eureka. Whereas both machines routinely exercise the path to the file servers in the course of conventional parallel and serial I/O, the switching fabric that enables these paths also offers the possibility for processes on Intrepid to speak directly with processes on Eureka. Preparatory to building and testing new APIs to enable these high-speed communications, we are testing parallel streaming along that path.

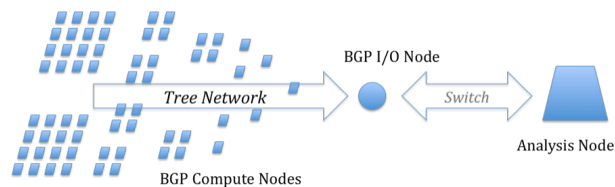


Figure 5. System setup for the internal network experiments.

Figure 5 depicts the network architecture for communication between a Blue Gene/P compute node and a data analysis node. There are 64 compute nodes in a partition and a single BG/P I/O node (ION) for each partition. The compute nodes are connected

to the ION via a tree network with a bandwidth of 6.8 Gbps between the compute nodes and the ION. The ION is connected to the analysis node on a 10 Gbps network.

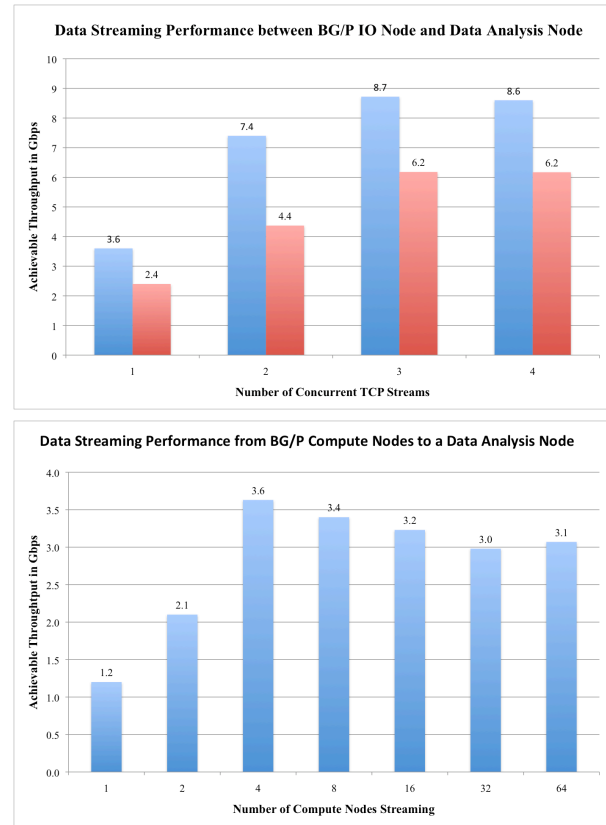


Figure 6. Results for streaming between Intrepid (BG/P) and Eureka. At top are results in each direction between a single BG/P ION and a Eureka node (DA). At bottom are the results from a BG/P compute node (behind the ION) to a Eureka node.

We have conducted initial bidirectional streaming experiments between the ION and analysis node. From Figure 6 (top), we see that as we increase the number of streams, the achievable throughput increases. Increasing the streams beyond 4, however, does not yield any significant improvement and could actually lead to performance degradation because of the contention between the various streams.

In order to stream data from the compute node to the data analysis nodes, the compute node kernel forwards the system calls to the I/O node. The ION then forwards the data onto the analysis node using the CIOD daemon on the ION. As seen from Figure 6 (bottom), we are able to sustain a maximum bandwidth of 3.6 Gbps between a compute node partition and a data analysis node. The approach results in 52% network utilization and is limited by overhead of the forwarding in the ION.

Figure 7 depicts the performance of streaming as we scale the number of Blue Gene nodes from 128 nodes (512 cores) to 4096 nodes (16K cores). Note the linear scaling in the performance of streaming with the number of nodes. We are able to sustain 170

Gbps on 4K nodes. This would enable one to stream the 42 GB checkpoint data of the astrophysics application example from Intrepid to Eureka in 2 seconds, much faster than current time required to write the same data from Intrepid to storage.

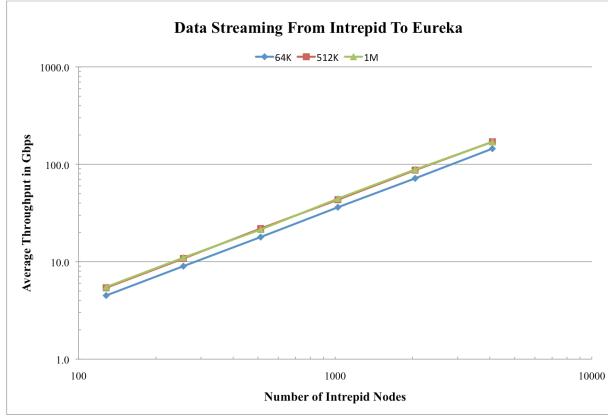


Figure 7. Performance of multistream experiments from Intrepid (BG/P) to Eureka.

3.4 Wide Area Experiments

And, finally, we are measuring and testing scenarios like that introduced in Figure 3(d). Figure 8 lays out some of the particulars of a recent test in which large datasets produced by a simulation run on the Kraken supercomputer at the National Institute for Computational Science at the University of Tennessee was transferred to Argonne National Laboratory via a 10 Gbps higher speed network over ESNet. They were then rendered at interactive rates on the Eureka cluster at ANL and streamed live to a large tiled display on the Supercomputing 2009 show floor in Portland, OR.

Each simulation timestep created a 4K by 4K by 4K uniform grid with several model variables in each of these 64 billion cells. One four-byte variable, the density of ordinary matter in this cosmology simulation, represented 256 GB of data. The rate that these single time step chunks of data were produced by the simulation was far slower than interactive – numbering several frames per day. With two 10 Gbps links between Tennessee and Illinois, that single timestep of one variable would take at least 200 seconds.

We had much better results using available bandwidth to stream rendered results. For this experiment we added multi-streaming to our v13 rendering application. The 4K by 2K rendered interactive stream was split into smaller tiles and shipped to Oregon over a 10 Gbps link. The link was generally fully subscribed by the multi-stream while v13 generating a few frames per second of live high-resolution volume renderings. Although the control stream for interaction was actually created at Eureka, implementing a light control stream from the remote site would not be difficult.

4. Conclusions

In this paper we have described a range of circumstances of critical interest to ultrascale computing which rely on efficient collaboration between processing subsystems. We’ve dubbed these *resource-coupled computations*. In situ methods couple

simulation and analysis in real time on a single shared compute node. Remote visualization couples high-performance compute and data resources to remote display clients. Co-analysis couples two high-performance compute resources in real time.

We described details of one application’s dataset footprint to illustrate opportunities for better overlapping of simulation and I/O, offloading I/O to a co-processing platform, and real time data analysis through co-processing. We then discussed several components of our measurement effort – measurements of coarse grain characteristics of simulation and analysis jobs running on ALCF resources, synthesis of these measurements into project footprints, early results from new streaming experiments that connect the BG/P simulation resource to the analysis and visualization cluster, and wide-area experiments.

Several opportunities may become feasible with co-processing enabled by fast data transfer from the simulation to a high-performance analysis cluster: mapping the Intrepid output data flow to Eureka may speed up some applications; offloading data organization and disk writes; transparent (and relatively free) co-analysis; and data compression. As an accelerator and co-analysis engine, a companion cluster such as Eureka might be considered inexpensive at only 1-2% of the cost of a large supercomputer such as Intrepid.

The work described in this paper aims to provide a system level understanding of data exchange between collaborating computational resources. It is informed by real computational workloads across resources spanning the entire scientific workflow. Our hope is to identify and develop new methods for designing workflows that more effectively utilize available computational resources by optimizing the end-to-end process.

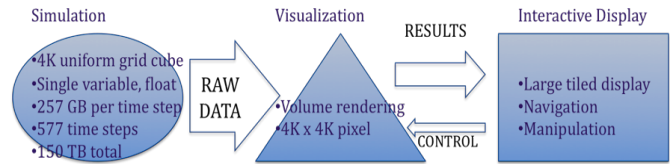


Figure 8. Schematic of the Stargate demo at Supercomputing 2009.

5. ACKNOWLEDGMENTS

This work was supported by a Director’s Postdoctoral Fellowship (V.V.); by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357; and by TeraGrid, under National Science Foundation Grant OCI-0504086. For helpful discussions we acknowledge Tom Peterka, Rob Ross, Narayan Desai, David Ressler, and William Scullin. For significant contributions to the wide area interactive visualization experiments we acknowledge Rick Wagner, Loren Wilson, Linda Winkler, Michael Norman, Robert Harkess, Larry Smarr, and ESNET2.

6. REFERENCES

- [1] Abbasi, H., Wolf, M., and Schwan, K. LIVE data workspace: A flexible, dynamic and extensible platform for petascale applications. In Cluster Computing, 2007 IEEE International Conference on, pages 341–348, Sept. 2007.

- [2] Foster, I., Geisler, J., Gropp, W., Karonis, N., Lusk, E., Thiruvathukal, G., and Tuecke, S. Wide-area implementation of the message passing interface. *Parallel Comput.*, 24(12-13):1735–1749, 1998.
- [3] Insley, J. A., Papka, M.E, Dong, S., Karniadakis, G., and Karonis, N. Runtime visualization of the human arterial tree. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):810–821, 2007.
- [4] Johnsen, E., Bjørndalen, J.M., and Anshus, O.J. CoMPI - configuration of collective operations in LAM/MPI using the scheme programming language. In Bo Kågström, Erik Elmroth, Jack Dongarra, and Jerzy Wasniewski, editors, *PARA*, volume 4699 of *Lecture Notes in Computer Science*, pages 189–197. Springer, 2006.
- [5] Krishnaprasad, N., Vishwanath, V., Venkataraman, S., Rao, A., Renambot, L., Leigh, L., Johnson, A.E, and Davis, B. Juxtaview - a tool for interactive visualization of large imagery on scalable tiled displays. In *CLUSTER*, pages 411–420, 2004.
- [6] Kohl, J., Wilde, T., and Bernholdt, D. Cumulvs: Interacting with high-performance scientific simulations, for visualization, steering and fault tolerance. *Int. J. High Perform. Comput. Appl.*, 20(2):255–285, 2006.
- [7] Larson, J., Jacob, R., and Ong, E. The model coupling toolkit: A new fortran90 toolkit for building multiphysics parallel coupled models. *Int. J. High Perform. Comput. Appl.*, 19(3):277–292, 2005.
- [8] Ma, K. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, 2009.
- [9] Vishwanath, V., Burns, R., Leigh, J., and Seablom, S. Accelerating tropical cyclone analysis using lambdaram, a distributed data cache over wide-area ultra-fast networks. *Future Generation Comp. Syst.*, 25(2):184–191, 2009.
- [10] Yu, H., Wang, C., Grout, R., Chen, J., and Ma, K. A study of in situ visualization for petascale combustion simulations. Technical Report CSE-2009-9, University of California at Davis, 2009.

This government license must be at the end of the paper for review submission, but removed before publication.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.